Modern Engineering

January 24 | Day 11

Prudential Financial

GENERAL ASSEMBLY

Whilst we wait....

In chat write:

Your name - role and a recent impulse buy.

LESSON ROADMAP

Mocks, Stubs, and Fakes

Test Driven Development Lab

Jest Testing Suites

UI Testing with Jest

Testing a React App

MEF MODULE 4 DAY 11: UI Tests with Jest

Schedule	
9:00–9:15 am	Welcome and Warm-Up
9:15–9:45 am	**Mocks, Stubs, and Fakes**
9:45 am–12:30 pm	**Test Driven Development in Action Lab**
12:30–1:30 pm	Lunch
1:30-2:30 pm	**Testing React with Jest**
2:30–4:50 pm	**Adding Tests to the To Do React App**
4:50–5:00 pm	Bring It Home

LEARNING OBJECTIVES

1. Identify the challenges of integration tests and using external resources in testing
2. Design integration tests in an express API
3. Structure test suites in the Jest library
4. Use Jest to test the UI components of a react application

Modern Engineering

Mock, Stubs, Fakes

Mocks Stubs and Fakes

Mocks, stubs, and fakes are terminology used in test-driven development (TDD) to describe objects that mimic the behavior of real objects.

To guarantee that the tests are predictable and repeatable, these objects are used to isolate the unit of code being tested. What each phrase means is as follows:

- **Mocks**: Replicates real-world data
- **Stubs**: Produces pre-configured responses
- **Fakes**: A stripped-down version of a real object

Mocks

Mocks are objects that simulate the behaviors and characteristics of real objects, including their methods. They are frequently used to make assertions regarding the actual usage of the things.

Mocks can be used to check that the code being tested makes the required calls and passes the correct parameters by being programmed to return particular values or throw exceptions when they are called.

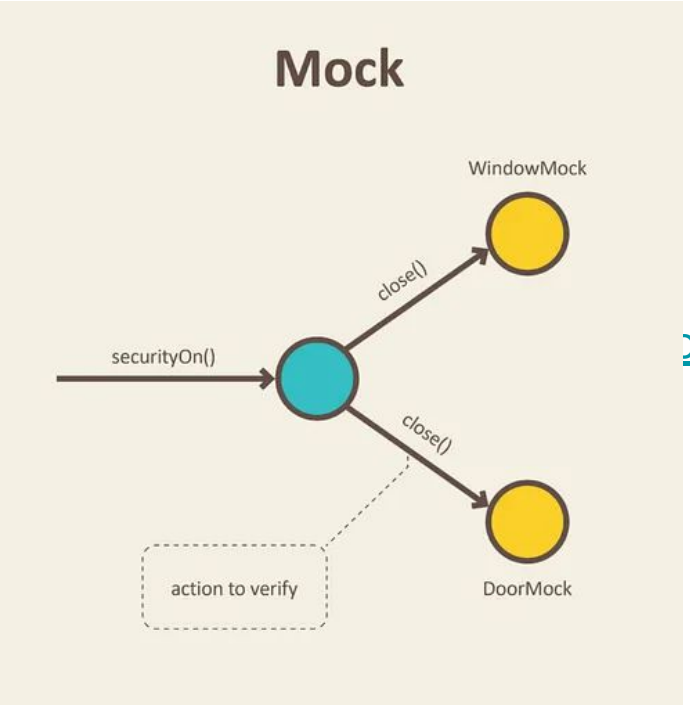

Image 9: Mocks Diagram

Stubs

Similar to mocks, **stubs only substitute specific methods or properties of the real objects**. However, they are often simpler.

They are used to regulate the actions of the actual objects and guarantee that the tests are independent of outside dependencies.

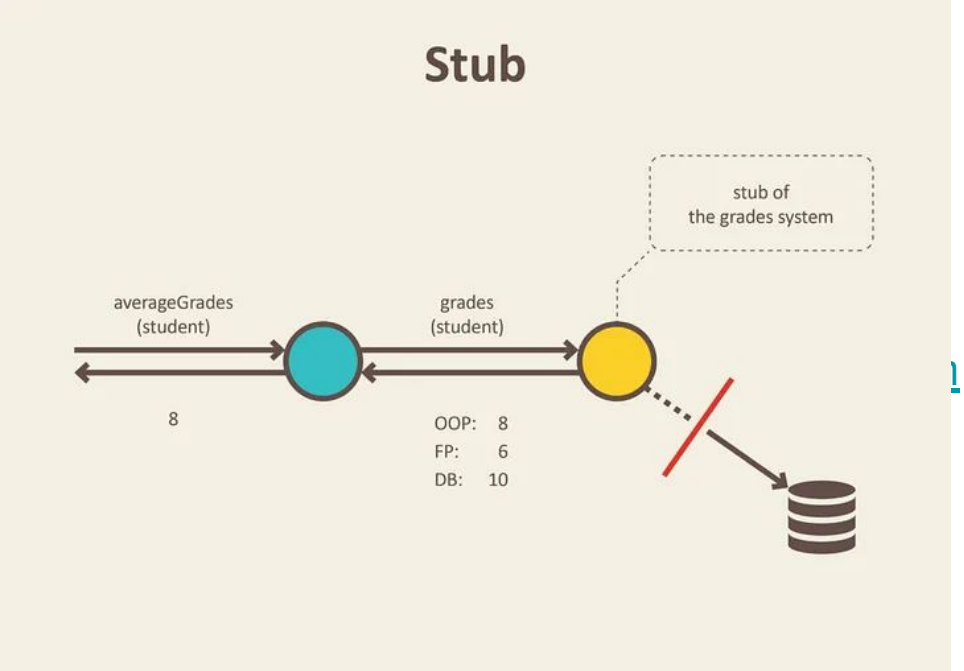

Image 10: Stubs Diagram

Fakes

Fakes are things that simulate the behavior of real objects but don't really use those things' implementation. Instead, **they offer a simplified implementation that is intended only for testing**.

When the true implementation of an object is too complex to test effectively or when there are performance or reliability problems with the actual implementation, fakes are frequently utilized.

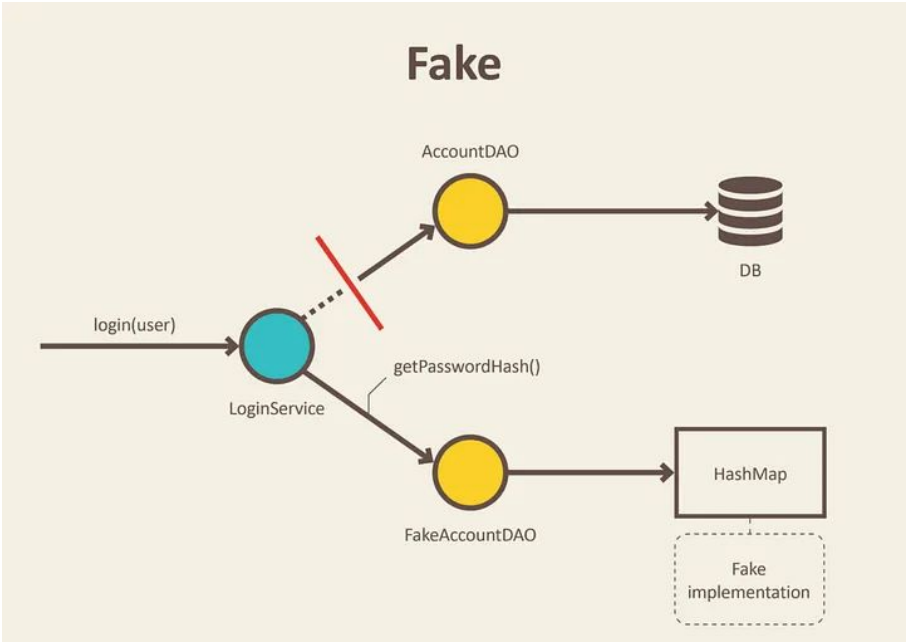

Image 11: Fakes Diagram

Risks of Stale Mocks

In test-driven development (TDD), stale mocks can provide a number of risks that can negatively impact both the software development process and overall program quality. Among these risks are:

- Inconsistent Behavior
- Imprecision
- Reduced test coverage
- Decreased confidence in test results
- Increased Maintenance costs

UI Unit Tests

- UI unit tests are used to test modules that deal with user input and behavior of the UI.
    - You can use the same test frameworks you use on a node.js backend for most UI unit testing
- UI unit testing can be used to test UI components like buttons, menus, carousels, etc. various behavior's that will be triggered by user input, usually from a keyboard or mouse.
- Commonly uses headless browsers.
- Not the same as writing unit test for your services and components on a front-end app.

UI Unit Tests: Pros and Cons

Pros

- Provides test coverage of UI components
- Provided declarative tests for UI behavior
- Reducing chance of UI bugs

Cons

- Not universally useful
- 100% UI test coverage is even more debatable in value
- Time demanding

Resilience testing

- Resilience testing is a **family of tests** that evaluate how well a system will operate under stress and in chaotic situations while still remaining as usable as possible.
- Some situations resilience testing is interested in:
  - Service outages
  - Extreme load
  - Network failures
  - Crashes

Discussion:

How much testing is enough?

10 Minutes

Testing can vary depending on the project, the stakeholders, and the purpose of the application you are developing. Let's explore a question together.

How do you know when you've done **enough** testing for your project?

Come off mute and discuss!

Solo Exercise:

Test Driven Development In Action - LOTR

120 minutes

We've created a basic express API with a full suite of tests: but none of them pass!

Your goal in this LOTR lab is to read the tests carefully (you won't be provided any further instructions) to deduce what code you need to add to the API in order to make them pass.

You should not be writing any new testing code.

Break Time

Modern Engineering

Jest with React

Guided Walk-Through:
Using Jest to Test React

Let's see how to test our react applications using the Jest library by following the walkthrough in this Github repository

Solo Exercise:

Test the To Do List React App

120 minutes

Use Jest and the react Testing Library to implement testing in the react To Do list application, following the directions and starter code provided in this git repository.

Solo Exercise:

Daily Exit Ticket

3 minutes

- Please take a moment to give us your feedback after today's course!
- Use the QR code or follow the link: https://bit.ly/pruMEFc2

Scan Me!


GA